Diese Kopfleiste bitte unbedingt ausfüllen!

Familienname, Vorname (bitte durch eine Leerspalte trennen)

Bereich	Berufsnummer	IHK-Nummer	Prüflingsnummer
6 7	1 2 0 1		
Sp. 1-2	Sp. 3-6	Sp. 7-9	Sp. 10-14

Termin: Mittwoch, 23. November 2022

Abschlussprüfung Winter 2022/23

1201

2 Entwicklung und Umsetzung von Algorithmen

Fachinformatiker
Fachinformatikerin
Anwendungsentwicklung (AO 2020)

Teil 2 der Abschlussprüfung

4 Aufgaben
mit Belegsatz
90 Minuten Prüfungszeit
100 Punkte

Bearbeitungshinweise

1. Bevor Sie mit der Bearbeitung der Aufgaben beginnen, überprüfen Sie bitte die **Vollständigkeit** dieses Aufgabensatzes. Die Anzahl der zu bearbeitenden Aufgaben ist auf dem Deckblatt links angegeben. Wenden Sie sich bei Unstimmigkeiten sofort an die Aufsicht, weil Reklamationen am Ende der Prüfung nicht anerkannt werden können.
2. Füllen Sie zuerst die **Kopfzeile** aus. Tragen Sie Ihren Familiennamen, Ihren Vornamen und Ihre Prüflings-Nr. in die oben stehenden Felder ein.
3. Lesen Sie bitte den **Text** der Aufgaben ganz durch, bevor Sie mit der Bearbeitung beginnen.
4. Halten Sie sich bei der Bearbeitung der Aufgaben genau an die **Vorgaben der Aufgabenstellung** zum Umfang der Lösung. Wenn z. B. vier Angaben gefordert werden und Sie sechs Angaben anführen, werden nur die ersten vier Angaben bewertet.
5. Tragen Sie die frei zu formulierenden **Antworten dieser offenen Aufgaben** in die dafür lt. Aufgabenstellung vorgesehenen Bereiche (Lösungszeilen, Formulare, Tabellen u. a.) des Arbeitsbogens ein.
6. Sofern nicht ausdrücklich ein Brief oder eine Formulierung in ganzen Sätzen gefordert werden, ist eine **stichwortartige Beantwortung** zulässig.
7. Schreiben Sie deutlich und gut lesbar. Ein nicht eindeutig zuzuordnendes oder **unleserliches Ergebnis** wird als **falsch** gewertet.
8. Zur Lösung der Rechenaufgaben darf ein nicht programmierter, netzunabhängiger **Taschenrechner** ohne Kommunikationsmöglichkeit mit Dritten verwendet werden.
9. Wenn Sie ein **gerundetes Ergebnis** eintragen und damit weiterrechnen müssen, rechnen Sie (auch im Taschenrechner) nur mit diesem gerundeten Ergebnis weiter.
10. Für **Hilfsaufzeichnungen** können Sie das in der Tasche beigelegte Konzeptpapier verwenden. Bewertet werden jedoch grundsätzlich nur Ihre Eintragungen in diesem Aufgabensatz.

Wird vom Korrektor ausgefüllt!

Bewertung

Für die Bewertung gilt die Vorgabe der Punkte in den Lösungshinweisen.

1. Aufg. ☐☐ Punkte (15 16) 2. Aufg. ☐☐ Punkte (17 18) 3. Aufg. ☐☐ Punkte (19 20) 4. Aufg. ☐☐ Punkte (21 22)

Prüfungszeit ☐ (23)
Die entsprechende Ziffer (1, 2 oder 3) finden Sie in der Abfrage nach der Prüfungszeit im Anschluss an die letzte Aufgabe.

Gesamtpunktzahl ☐☐☐ (24 25 26)

Prüfungsort, Datum

Unterschrift

Gemeinsame Prüfungsaufgaben der Industrie- und Handelskammern. Dieser Aufgabensatz wurde von einem überregionalen Ausschuss, der entsprechend § 40 Berufsbildungsgesetz zusammengesetzt ist, beschlossen. Hinweis: Im Interesse einer besseren Lesbarkeit wird in der Aufgabenstellung und in den Angaben zur Aufgabenstellung nur die männliche Form (generisches Maskulinum) verwendet. Die verkürzte Sprachform beinhaltet keine Wertung und die gewählten männlichen Formulierungen gelten uneingeschränkt auch für die weiteren Geschlechter.
Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2022 – Alle Rechte vorbehalten!

Die Aufgaben 1 bis 4 beziehen sich auf die folgende Ausgangssituation:

Ein bundesweit tätiger Bildungsträger bietet an acht attraktiven Standorten Tagungen, Seminare und Veranstaltungen an. An den Standorten stehen insgesamt 45 Seminarräume mit mehr als 600 IT-Arbeitsplätzen zur Verfügung. Die Trainings sind auch als Online-Trainings buchbar.

Alle IT-Services werden von zwei Rechenzentren ausgeführt.

Da derzeitig nicht alle Schulungs- und Büroräume ausgelastet sind, sollen zukünftig die Raumressourcen den Kunden für individuelle Bedürfnisse zur Verfügung gestellt werden.

Korrekturrand

1. Aufgabe (25 Punkte)

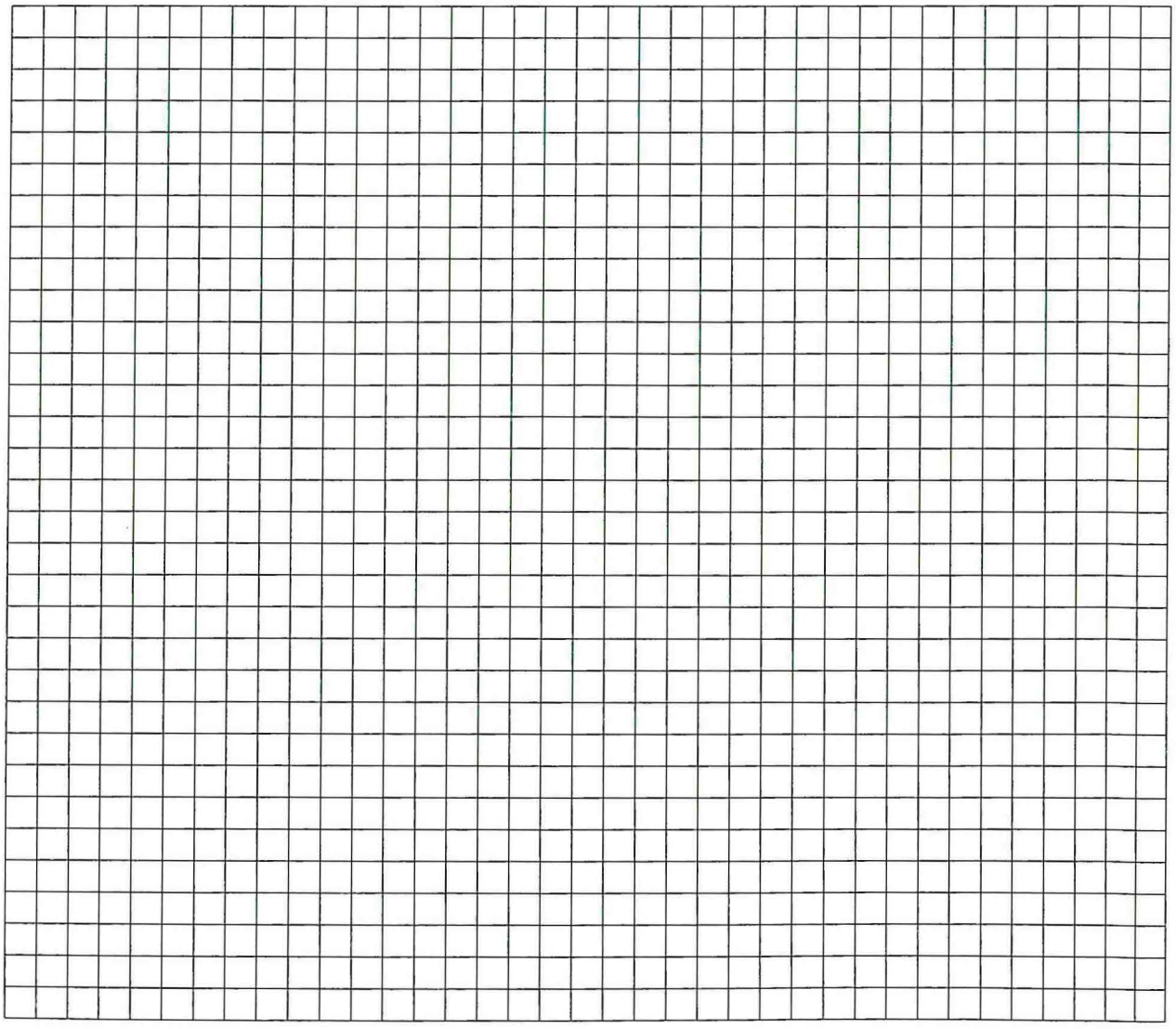
Im Rahmen der Erstellung von Lernsoftware soll zunächst ein Sortieralgorithmus für ein Array von Raum-Objekten erstellt werden. Die Sortierung soll aufsteigend nach der Belegungszahl eines Raumes erfolgen.

Raum
...
`+ getBelegung(): Integer` ...

Der Sortieralgorithmus kann beispielsweise nach folgendem Verfahren arbeiten:
- Vergleich der ersten beiden Elemente, gegebenenfalls tauschen
- Vergleich des 2. und 3. Elementes, gegebenenfalls tauschen
- Fortführen des n. und n+1. Elementes bis zum Ende des Arrays
- Wiederholen, bis das Array sortiert ist

a) Erstellen Sie einen Algorithmus, der ein Array von Raum-Objekten nach dem genannten oder einem alternativen Verfahren sortiert. Geben Sie den Algorithmus in Pseudocode, Struktogramm oder als PAP an. 10 Punkte

```
void sort (raeume: Array von Raum)
```



Korrekturrand

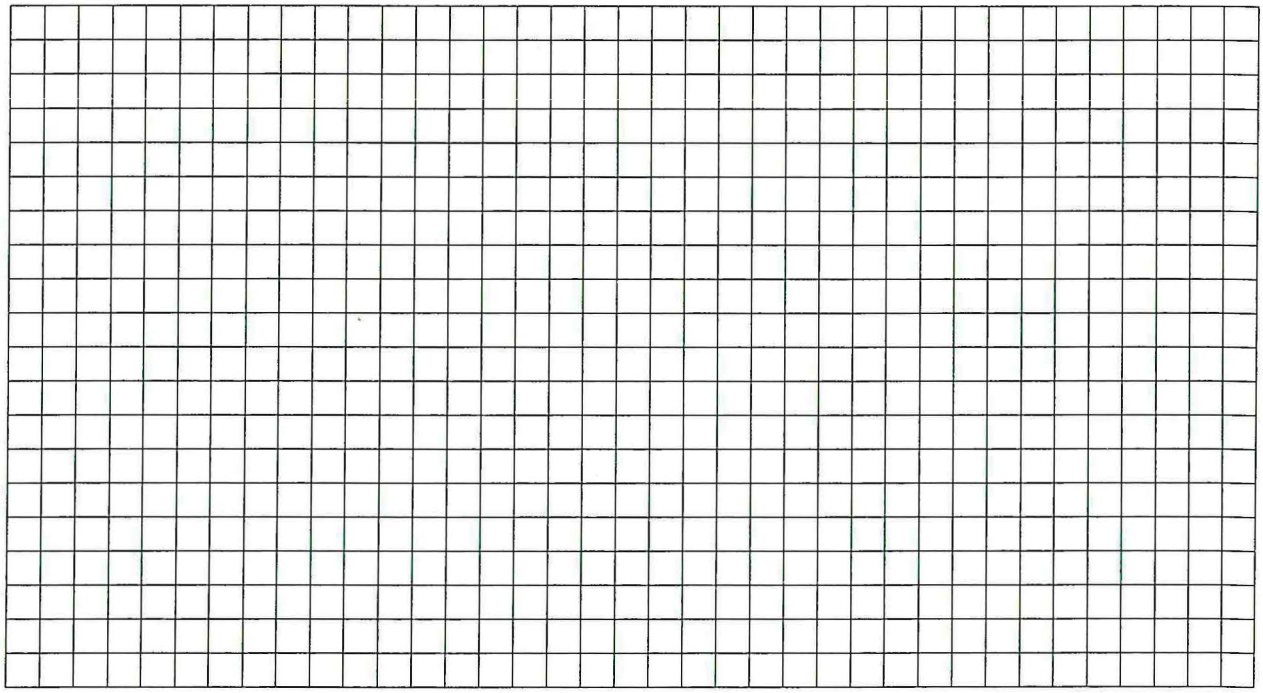
b) Zur Flexibilisierung des Sortieralgorithmus wird eine Funktion benötigt, die …
- den Wert 0 zurückgibt, wenn die zwei der Funktion übergebenen Objekte gleich („groß") sind,
- einen positiven Wert zurückgibt, wenn der erste Parameter größer als der zweite ist und
- einen negativen Wert im verbleibenden Fall.

Erstellen Sie eine Funktion vergleicheMitarbeiter(Mitarbeiter m1, Mitarbeiter m2), die zwei Mitarbeiter-Objekte als Parameter bekommt und einen entsprechenden Rückgabewert liefert. Der Vergleich der Mitarbeiter soll über das Gehalt vorgenommen werden.

Geben Sie den Algorithmus in Pseudocode, Struktogramm oder als PAP an. 5 Punkte

Mitarbeiter
…
+ getGehalt(): Double …

`vergleicheMitarbeiter(Mitarbeiter m1, Mitarbeiter m2): Integer`



c) Da nicht nur Raum-Objekte sortiert werden sollen, wird zur Erstellung eines flexibleren Algorithmus die Verwendung von generischen Klassen vorgeschlagen.

Erläutern Sie den Begriff generische Klasse. 3 Punkte

Fortsetzung 1. Aufgabe →

Fortsetzung 1. Aufgabe

Korrekturrand

d) Zur Erstellung eines verbesserten Sortieralgorithmus soll die vorhandene generische Klasse Function<T> verwendet werden. Ein Function-Objekt kann eine Funktion speichern, die zwei Parameter vom Typ T erwartet und einen Integerwert zurückgibt.

Verwendung eines Function<T>-Objektes (Beispiel):

Speichern einer Funktion in einem Function-Objekt:	**Function<Mitarbeiter> f = vergleicheMitarbeiter;**
Aufruf der gespeicherten Funktion:	**f(mitarbeiter1, mitarbeiter2)**

Zusätzlich soll für den neuen Sortieralgorithmus die vorhandene generische Klasse List<T> verwendet werden, die Objekte von Typ T speichert.

List<T>
...
+ get(index: Integer) : T + set(index: Integer, element: T) : void + size() : Integer ...

Aktualisieren Sie Ihren Sortieralgorithmus so, dass dieser als Parameter ein List<T>-Objekt und ein Function<T>-Objekt erhält. Für den Vergleich im Sortieralgorithmus soll die im Function<T>-Objekt gespeicherte Funktion benutzt werden.

7 Punkte

```
sort(List<T> liste, Function<T> f): void
```

2. Aufgabe (25 Punkte)

Korrekturrand

Für einen Kurs sollen Online-Lehrmittel erstellt werden. Sie erhalten die Aufgabe, einen vorgegebenen rekursiven Algorithmus zur Umrechnung einer Dezimalzahl in eine Binärzahl mit einem Schreibtischtest auf Korrektheit zu prüfen und gegebenenfalls zu korrigieren.

a) Folgender Algorithmus in Pseudocode ist vorgegeben:

```
void Umrechnung(dezimalzahl: int)
BEGINN
      rest: int;
      rest = dezimalzahl modulo 2;
      AUSGABE(rest);
      dezimalzahl = dezimalzahl div 2; //div ist Division
      WENN dezimalzahl > 0 DANN
      Umrechnung(dezimalzahl);
      ENDE
ENDE
```

aa) Führen Sie den Schreibtischtest aus und tragen Sie das Ergebnis in die rechte Spalte „Ergebnis Schreibtischtest" ein. 4 Punkte

Testfall	Dezimalzahl	Erwartetes Ergebnis	Ergebnis Schreibtischtest
1	7	111	
2	11	1011	

ab) Beschreiben Sie den Fehler und eine Korrekturmöglichkeit. 4 Punkte

ac) Der Algorithmus soll in einem White-Box-Test geprüft werden.

Erklären Sie den Begriff Anweisungsüberdeckung und bestimmen Sie die mindestens benötigte Anzahl an Testfällen, um bei dem gegebenen Algorithmus Anweisungsüberdeckung zu erreichen. 3 Punkte

b) Beschreiben Sie je einen Vor- und Nachteil der rekursiven Lösung im Vergleich zu einer iterativen Lösung. 4 Punkte

Vorteil:

Nachteil:

Fortsetzung 2. Aufgabe →

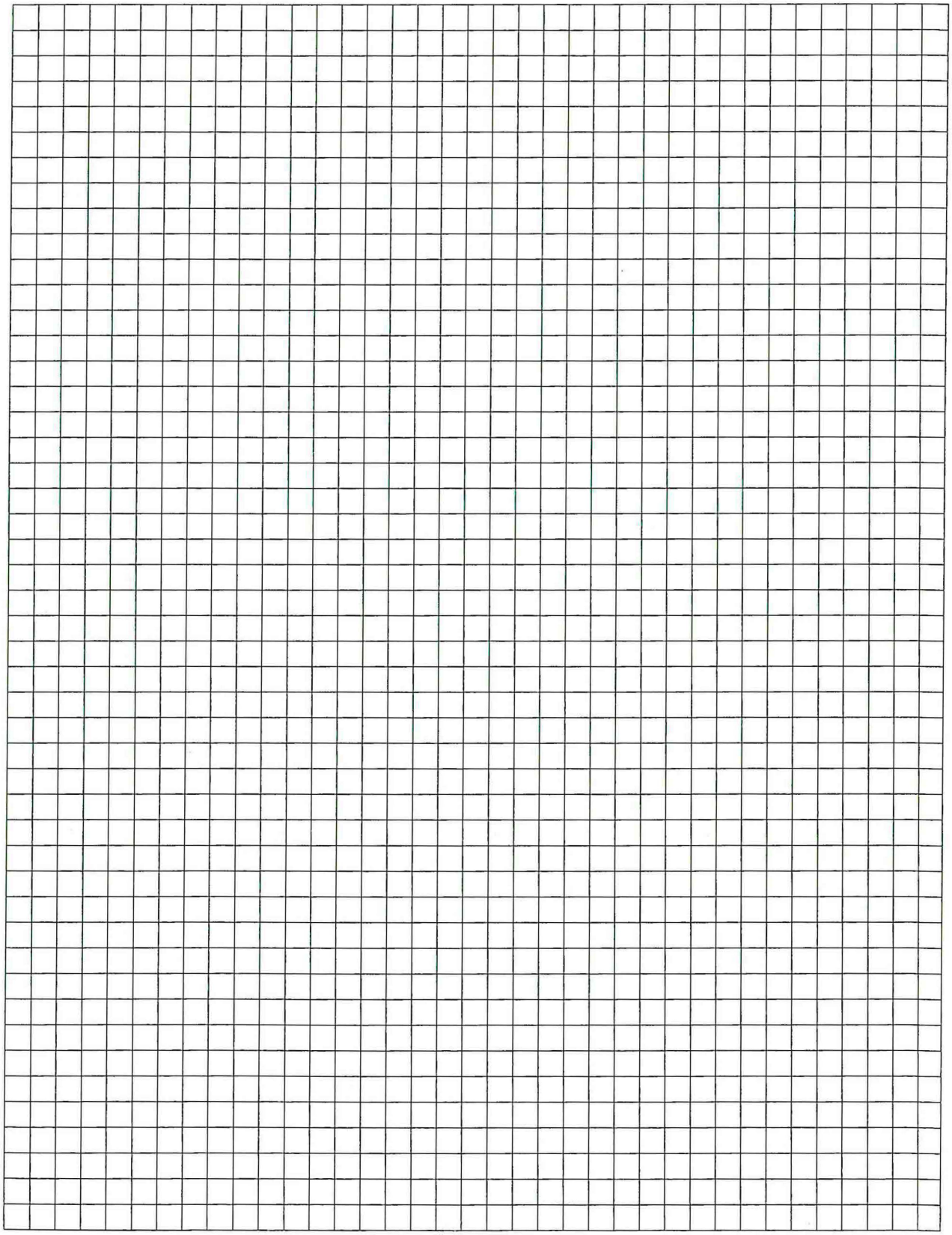
Korrekturrand

c) Der rekursive Algorithmus soll durch einen iterativen Algorithmus ersetzt werden. Eine Klasse `List` kann verwendet werden. Folgende Methoden der Klasse List stehen zur Verfügung:

`insert(Wert:int):void`	Fügt den Wert „Wert" am Ende der Liste ein
`size():int`	liefert die Anzahl der Listenelemente
`element(Pos:int):int`	Liefert das Listenelement an Position Pos, Position 0 ist das Listenende
`List()`	Konstruktor erzeugt eine leere Liste

Erstellen Sie einen iterativen Algorithmus, der die Umrechnung korrekt ausführt. 10 Punkte

```
void Umrechnung(dezimalzahl: int)
```



Dieses Blatt kann an der Perforation aus dem Aufgabensatz herausgetrennt werden!

Abbildung zur 3. Aufgabe

Buchungsbestätigung

Buchungsnummer.: 103864
Datum der Buchung: 19. September 2022

Max Mustermann
Hauptstraße 23
01129 Dresden
Tel: 0123456789
E-Mail: max@mustermann.de

Kundennummer: 122
Ansprechpartner: Christine Wald
Durchwahl: 012345-4789
E-Mail: cwald@bt-abc.de

Sehr geehrter Herr Mustermann,

vielen Dank für Ihre Buchung. Nachfolgend finden Sie alle Informationen zu Ihrer Buchung:

Standort: Berlin Mietzeitraum: 4. Oktober 2022 bis 7. Oktober 2022

Bildungsträger ABC
Wasserstraße 1
11111 Berlin

Räume am Standort:
1. Raum: 201 Art: Tagungsraum
2. Raum: 202 Art: Tagungsraum
3. Raum: 304 Art: Schulungsraum

Ressourcen:
1 x Laserdrucker
3 x Beamer
2 x Flipchart
1`x mobiles Whiteboard

Bei Fragen zu Ihrer Buchung wenden Sie sich direkt an unsere Mitarbeiterin Frau Wald, welche diese Buchung betreut.

Mit freundlichen Grüßen
Ihr Serviceteam

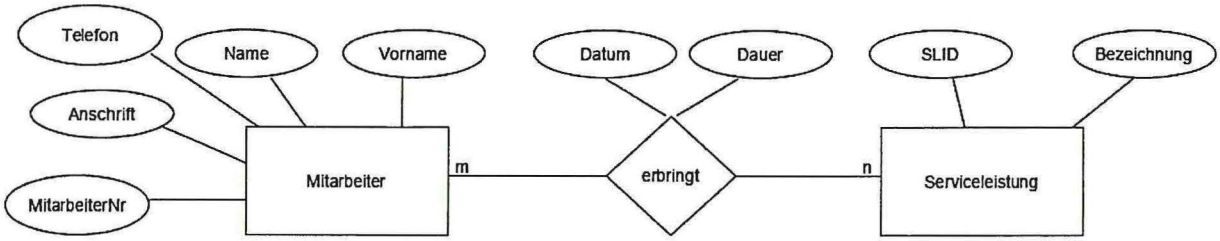
Zusätzliche Anforderungen:
- Die Buchung soll in Verbindung zum betreuenden Mitarbeiter stehen.
- Die Standorte der Räume sollen auch erfasst werden.

3. Aufgabe (25 Punkte)

Korrekturrand

Wichtige Daten der Firma sollen in einer relationalen Datenbank gespeichert werden. Dazu liegt ein erstes ER-Modell vor, in dem schon die Entitätstypen Mitarbeiter und Serviceleistungen erfasst wurden. Dieses Modell soll nun um die Raumbuchungen von Kunden erweitert werden. Als Grundlage dafür steht Ihnen die nachfolgende Buchungsbestätigung und die danach aufgeführten Anforderungen zur Verfügung.

a) Erweitern Sie das nachstehende ER-Modell zunächst nur um die notwendigen Entitätstypen und tragen Sie die Beziehung zwischen diesen mit den entsprechenden Kardinalitäten ein. (Hinweis: Attribute müssen nicht eingetragen werden.) 12 Punkte



Fortsetzung 3. Aufgabe →

Fortsetzung 3. Aufgabe

Korrekturrand

b) Aus dem ER-Modell soll im Anschluss ein relationales Datenmodell erstellt werden.

Überführen Sie dazu **nur** die (in Aufgabe 3 a)) gegebenen **Entitätstypen Mitarbeiter und Serviceleistung**, deren Beziehung und alle Attribute in ein relationales Datenmodell, welches der 3. Normalform genügt.

Kennzeichnen Sie alle Primärschlüssel mit PK und alle Fremdschlüssel mit FK. Tragen Sie alle notwendigen Beziehungen mit den entsprechenden Kardinalitäten ein.

Hinweis: Die Daten der Anschrift müssen nicht in einer eigenen Tabelle ausgelagert werden. 10 Punkte

c) Bei der Eingabe des relationalen Datenmodells müssen allen Attributen Datentypen zugeordnet werden.

Legen Sie für nachfolgende Attribute sinnvolle Datentypen fest. 3 Punkte

Attribut	Datentyp
Dauer	
PLZ	
Name	

4. Aufgabe (25 Punkte)

Ein Auszug eines Datenbanksystems zu Mitarbeiterdaten liegt Ihnen wie folgt vor.

Mitarbeiter								
Ma_ID	**Nachname**	**Vorname**	**Strasse**	**PLZ**	**Ort**	**GebDat**	**ArbeitszeitTag**	**Abt_ID**
1	Meier	Wilhelm	Musterstr. 22	50733	Köln	23.11.1997	8	3
2	Franz	Anja	Hauptstr. 333	50737	Köln	02.02.1964	4	3
3	Schmitt	Andrea	Aachener Str. 5	50968	Köln	02.08.1996	8	2
4	Kreuzer	Peter	Hauptstr. 155	50734	Köln	05.09.1977	8	1
5	Lang	Ina	Kölner Str. 221	52063	Aachen	11.11.2001	8	3
6	Kurz	Katrin	Klichstr. 3	50969	Köln	03.01.1979	8	1
7	Pfennig	Petra	Von Bern Str. 5	50999	Köln	08.10.1956	6	1
8	Krenz	Karin	Perterstr.	50997	Köln	19.12.1988	8	2
...								

Abteilung	
Abt_ID	**Abteilung**
1	Verwaltung
2	Produktion
3	Entwicklung
...	

FehlzeitArt	
FzA_ID	**Fehlzeitart**
1	AU
2	Urlaub
3	Dienstreise
...	

Fehlzeit					
Fz_ID	**Ma_ID**	**FzA_ID**	**VonDat**	**BisDat**	**FehltageSum**
1	2	1	10.05.2022	20.05.2022	9
2	5	2	16.05.2022	29.05.2022	10
3	3	1	18.05.2022	22.05.2022	3
4	1	1	19.05.2022	22.05.2022	2
5	1	2	23.05.2022	29.05.2022	5
6	7	2	30.05.2022	19.06.2022	15
7	6	3	25.05.2022	26.05.2022	2
8	6	3	06.06.2022	08.06.2022	3
9	1	1	27.06.2022	29.06.2022	3
...					

Jahresübergreifende Fehlzeiten werden getrennt nach Jahr eingetragen.
Bei FehltageSum werden Wochenenden und freie Tage nicht als Fehltage gezählt.

a) Erstellen Sie eine SQL-Abfrage, mit der alle Mitarbeiter mit Abteilungen, den summierten Urlaubstagen sowie der Anzahl der Urlaubseinträge im Jahr 2022 ausgegeben werden. 10 Punkte

Die Ergebnistabelle soll wie folgt aussehen:

ID	Nachname	Vorname	Abteilung	Urlaubstage	Urlaubseintraege
1	Meier	Wilhelm	Entwicklung	5	1
2	Franz	Anja	Entwicklung		0
3	Schmitt	Andrea	Produktion		0
4	Kreuzer	Peter	Verwaltung		0
5	Lang	Ina	Entwicklung	10	1
6	Kurz	Katrin	Verwaltung		0
7	Pfennig	Petra	Verwaltung	15	1
8	Krenz	Karin	Produktion		0
...					

Fortsetzung 4. Aufgabe →

b) Erstellen Sie die notwendigen SQL-Anweisungen, mit der Sie mit den Daten aus der Tabelle Fehlzeit vom Mitarbeiter Wilhelm Meier (Ma_ID: 1) in die Tabelle Fehlzeit_Archiv verschieben (kopieren und löschen) können. 10 Punkte

Hinweis: Die Tabelle Fehlzeit_Archiv existiert und hat die gleiche Struktur wie die Tabelle Fehlzeit.

c) Erstellen Sie eine SQL-Abfrage, mit der alle Mitarbeiter ausgegeben werden, welche an der vierten Stelle der Postleitzahl (als Zeichenkette angelegt) die Ziffer 3 enthalten. 5 Punkte

Die Ergebnistabelle lautet wie folgt:

Ma_ID	Nachname	PLZ	Ort
1	Meier	50733	Köln
2	Franz	50737	Köln
4	Kreuzer	50734	Köln

PRÜFUNGSZEIT – NICHT BESTANDTEIL DER PRÜFUNG!

Wie beurteilen Sie nach der Bearbeitung der Aufgaben die zur Verfügung stehende Prüfungszeit?

[1] Sie hätte kürzer sein können. [2] Sie war angemessen. [3] Sie hätte länger sein müssen. ☐

Abschlussprüfung Winter 2022/23

IHK

Belegsatz

Fachinformatiker/Fachinformatikerin
Anwendungsentwicklung (AO 2020)
1201

Teil 2 der Abschlussprüfung

Der Belegsatz ist Grundlage für beide Prüfungsbereiche.

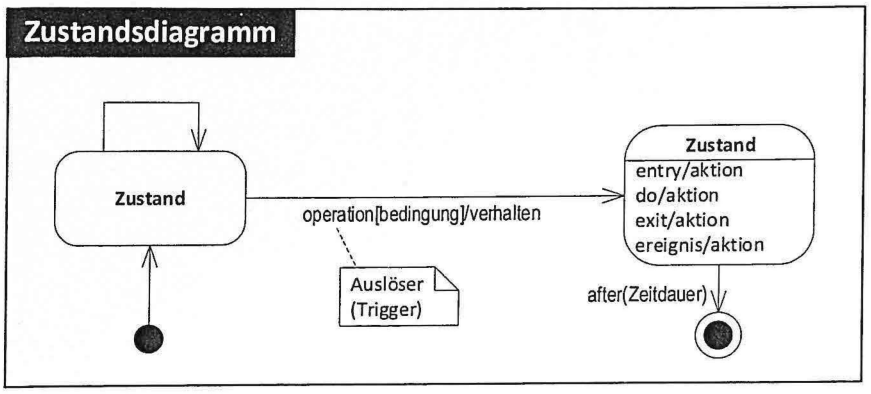
1. Planen eines Softwareproduktes
2. Entwicklung und Umsetzung von Algorithmen

	Seite
UML-Zustandsdiagramm	2
UML-Aktivitätsdiagramm	2
Pseudocode	3
DIN 66261 Struktogramm	4
SQL-Syntax (Auszug)	5/6

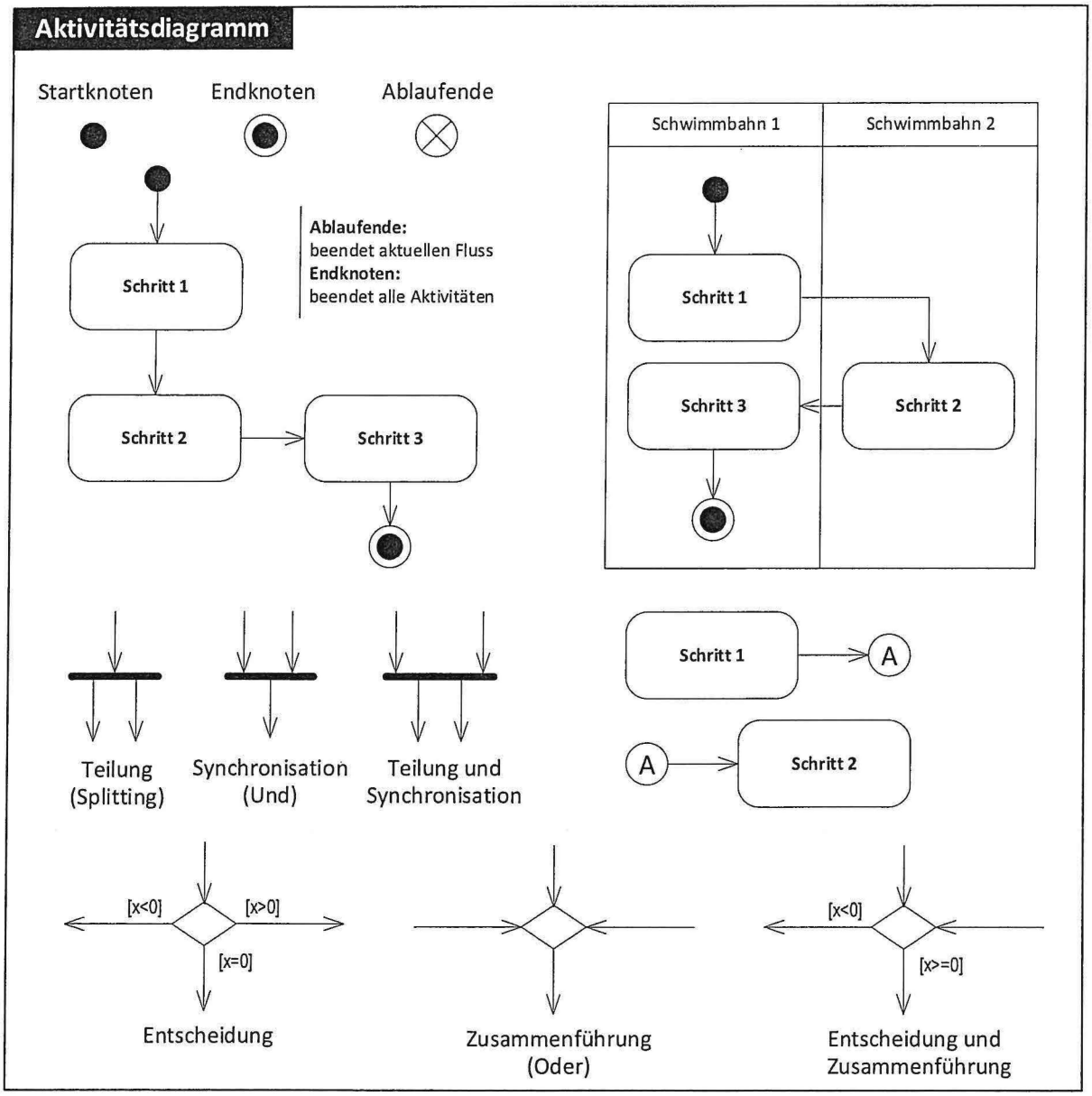
Gemeinsame Prüfungsaufgaben der Industrie- und Handelskammern. Dieser Aufgabensatz wurde von einem überregionalen Ausschuss, der entsprechend § 40 Berufsbildungsgesetz zusammengesetzt ist, beschlossen. Hinweis: Im Interesse einer besseren Lesbarkeit wird in der Aufgabenstellung und in den Angaben zur Aufgabenstellung nur die männliche Form (generisches Maskulinum) verwendet. Die verkürzte Sprachform beinhaltet keine Wertung und die gewählten männlichen Formulierungen gelten uneingeschränkt auch für die weiteren Geschlechter.
Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2022 – Alle Rechte vorbehalten!

2

UML-Zustandsdiagramm



UML-Aktivitätsdiagramm



Pseudocode

Die Darstellung soll in allgemein verständlichem Pseudocode erfolgen. Der Code soll für Dritte, ohne Kenntnis der verwendeten Programmiersprache, lesbar sein. Der Code muss nicht in der geschriebenen Sprache kompilierbar bzw. ausführbar sein.

Sie können gerne folgende Codefragmente verwenden:

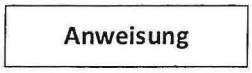
Pseudocode	Beschreibung

wenn [Bedingung] **dann**
  [Anweisung]
sonst
  [Anweisung]
Ende wenn
	Verzweigung

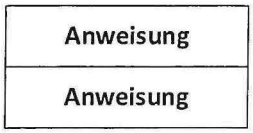
zähle [Variable] **von** [Startwert] **bis** [Endwert] [Schleifeninhalt] **Ende zähle**	Zählschleife (Iteration)
solange [Bedingung] [Schleifeninhalt] **Ende solange**	Kopfgesteuerte Schleife
wiederhole [Schleifeninhalt] **solange** < Bedingung >	Fußgesteuerte Schleife
:=	Zuweisung
=; <; >; <=; >=; < >	Vergleiche
integer	
double	
string	
array	

DIN 66261 Struktogramm

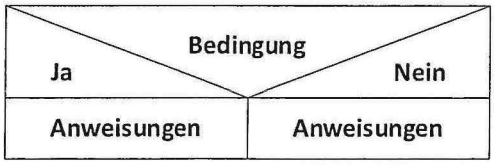
Struktogramm



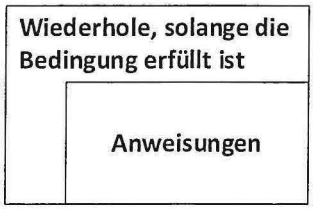
Verarbeitungsanweisung

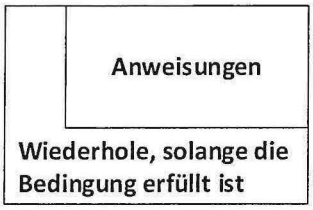


Sequenz (Folgestruktur), Anweisungen

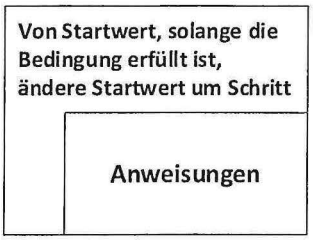


Bedingte Verzweigung (einseitige/zweiseitige Auswahlstruktur)

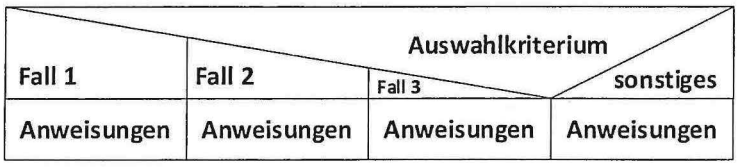




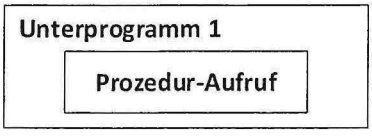
Wiederholstruktur (Schleife) **kopf**gesteuert und **fuß**gesteuert



Zählschleife



Fallauswahl (Mehrfach-verzweigung)



Unterprogramm

Wertzuweisungen werden durch := gekennzeichnet.

SQL-Syntax (Auszug)

Syntax	Beschreibung
Tabelle	
CREATE TABLE Tabellenname(Spaltenname < DATENTYP >, Primärschlüssel, Fremdschlüssel)	Erzeugt eine neue leere Tabelle mit der beschriebenen Struktur
ALTER TABLE Tabellenname **ADD COLUMN** Spaltenname **DROP COLUMN** Spaltenname **ADD FOREIGN KEY**(Spaltenname) **REFERENCES** Tabellenname(Primärschlüsselspaltenname)	Änderungen an einer Tabelle: Hinzufügen einer Spalte Entfernen einer Spalte Definiert eine Spalte als Fremdschlüssel
CHARACTER	Textdatentyp
DECIMAL	Numerischer Datentyp (Festkommazahl)
DOUBLE	Numerischer Datentyp (Doppelte Präzision)
INTEGER	Numerischer Datentyp (Ganzzahl)
DATE	Datum (Format DD.MM.YYYY)
PRIMARY KEY (Spaltenname)	Erstellung eines Primärschlüssels
FOREIGN KEY (Spaltenname) **REFERENCES** Tabellenname(Primärschlüsselspaltenname)	Erstellung einer Fremdschlüssel-Beziehung
DROP TABLE Tabellenname	Löscht eine Tabelle
Befehle, Klauseln, Attribute	
SELECT * | Spaltenname1 [, Spaltenname2, ...]	Wählt die Spalten einer oder mehrerer Tabellen, deren Inhalte in die Liste aufgenommen werden sollen; alle Spalten (*) oder die namentlich aufgeführten
FROM	Name der Tabelle oder Namen der Tabellen, aus denen die Daten der Ausgabe stammen sollen
SELECT ... **FROM ...** **(SELECT ...** **FROM ...** **WHERE ...) AS** tbl **WHERE ...**	Unterabfrage (subquery), die in eine äußere Abfrage eingebettet ist. Das Ergebnis der Unterabfrage wird wie eine Tabelle – hier mit Namen "tbl" – behandelt.
SELECT DISTINCT	Eliminiert Redundanzen, die in der Ausgabe auftreten können, Werte werden jeweils nur einmal angezeigt.
SELECT...INTO...FROM...WHERE	AAAAAAAAAAAAAAA.....
JOIN / INNER JOIN	Liefert nur die Datensätze zweier Tabellen, die gleiche Datenwerte enthalten
LEFT JOIN / LEFT OUTER JOIN	Liefert von der erstgenannten (linken) Tabelle alle Datensätze und von der zweiten Tabelle jene, deren Datenwerte mit denen der ersten Tabelle übereinstimmen
RIGHT JOIN / RIGHT OUTER JOIN	Liefert von der zweiten (rechten) Tabelle alle Datensätze und von der ersten Tabelle jene, deren Datenwerte mit denen der zweiten Tabelle übereinstimmen
WHERE	Bedingung, nach der Datensätze ausgewählt werden sollen
WHERE EXISTS (subquery) **WHERE NOT EXISTS (** subquery)	Die Bedingungen EXISTS prüft, ob die Suchbedingung einer Unterabfrage mindestens eine Zeile zurückliefert. NOT EXIST negiert die Bedingung.
WHERE ... IN (subquery) **WHERE NOT... IN (** subquery)	Der Wert des Datenfelds ist in der auswählten Menge vorhanden. Der Wert des Datenfelds ist in der auswählten Menge nicht vorhanden.
GROUP BY Spaltenname1 [,Spaltenname2, ...]	Gruppierung (Aggregation) nach Inhalt des genannten Feldes
ORDER BY Spaltenname1 [,Spaltenname2, ...] **ASC | DESC**	Sortierung nach Inhalt des genannten Feldes oder der genannten Felder ASC: aufsteigend; DESC: absteigend
LIMIT	Begrenzt die Anzahl der Ausgabezeilen

Fortsetzung SQL-Syntax →

Syntax	Beschreibung
Datenmanipulation	
DELETE FROM Tabellenname	Löschen von Datensätzen in der genannten Tabelle
UPDATE Tabellenname **SET**	Aktualisiert Daten in Feldern einer Tabelle
INSERT INTO Tabellenname[(spalte1, spalte2, ...)] **VALUES** (Wert für Spalte 1 [, Wert für Spalte 2, ...]) oder **SELECT** ... **FROM** ... **WHERE**	Fügt Datensätze in die genannte Tabelle, die entweder mit festen Werten belegt oder Ergebnis eines SELECT-Befehls sind
Berechtigungen kontrollieren	
CREATE Benutzer | Rolle **IDENTIFIED BY** 'Passwort'	Erzeugt einen neuen Benutzer oder eine neue Rolle mit einem Passwort
GRANT Recht | Rolle **ON** *.* | Datenbank.* | Datenbank.Objekt **TO** Benutzer | Rolle [WITH GRANT OPTION]	Weist einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt zu Weist einem Benutzer eine Rolle zu
REVOKE Rechte | Rollen **ON** *.* | Datenbank.* | Datenbank.Objekt **FROM** Benutzer | Rolle	Entzieht einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt Entzieht einem Benutzer eine Rolle
Aggregatfunktionen	
AVG(Spaltenname)	Ermittelt das arithmetische Mittel aller Werte im angegebenen Feld
COUNT(Spaltenname | *)	Ermittelt die Anzahl der Datensätze mit Nicht-NULL-Werten im angegebenen Feld oder alle Datensätze der Tabelle (dann mit Operator *)
SUM(Spaltenname | Formel)	Ermittelt die Summe aller Werte im angegebenen Feld oder der Formelergebnisse
MIN(Spaltenname | Formel)	Ermittelt den kleinsten aller Werte im angegebenen Feld
MAX (Spaltenname | Formel)	Ermittelt den größten aller Werte im angegebenen Feld
Funktionen	
LEFT(Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* der Zeichenkette von links.
RIGHT(Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* der Zeichenkette von rechts.
CURRENT	Liefert das aktuelle Datum mit der aktuellen Uhrzeit
CONVERT(time,[DatumZeit**])**	Liefert die Uhrzeit aus einer DatumZeit-Angabe
DATE(Wert)	Wandelt einen Wert in ein Datum um
DAY(Datum)	Liefert den Tag des Monats aus dem angegebenen Datum
MONTH(Datum)	Liefert den Monat aus dem angegebenen Datum
TODAY	Liefert das aktuelle Datum
WEEKDAY(Datum)	Liefert den Tag der Woche aus dem angegebenen Datum
YEAR(Datum)	Liefert das Jahr aus dem angegebenen Datum
DATEADD(Datumsteil, Intervall, Datum)	Fügt einem Datum ein Intervall (ausgedrückt in den unter Datumsteil angegebenen Einheiten) hinzu
DATEDIFF(Datumsteil, Anfangsdatum, Enddatum) Datumsteile: **DAY, MONTH, YEAR**	Liefert Enddatum-Startdatum (ausgedrückt in den unter Datumsteil angegebenen Einheiten)
Operatoren	
AND	Logisches UND
LIKE	Überprüfung von Text auf Gleichheit, wenn Platzhalter ("regular expressions") eingesetzt werden.
NOT	Logische Negation
OR	Logisches ODER
IS NULL	Überprüfung auf NULL
=	Test auf Gleichheit
>, >=, <, <=, < >	Test auf Ungleichheit
*	Multiplikation
/	Division
+	Addition, positives Vorzeichen
-	Subtraktion, negatives Vorzeichen

Stand 2021-09-30

Abschlussprüfung Winter 2022/23
Lösungshinweise

Fachinformatiker/Fachinformatikerin Anwendungsentwicklung (AO 2020) 1201

2 Entwicklung und Umsetzung von Algorithmen

Teil 2 der Abschlussprüfung

Allgemeine Korrekturhinweise

Die Lösungs- und Bewertungshinweise zu den einzelnen Handlungsschritten sind als Korrekturhilfen zu verstehen und erheben nicht in jedem Fall Anspruch auf Vollständigkeit und Ausschließlichkeit. Neben hier beispielhaft angeführten Lösungsmöglichkeiten sind auch andere sach- und fachgerechte Lösungsalternativen bzw. Darstellungsformen mit der vorgesehenen Punktzahl zu bewerten. Der Bewertungsspielraum des Korrektors (z. B. hinsichtlich der Berücksichtigung regionaler oder branchenspezifischer Gegebenheiten) bleibt unberührt.

Zu beachten ist die unterschiedliche Dimension der Aufgabenstellung (nennen – erklären – beschreiben – erläutern usw.).

Für die Bewertung gilt folgender Punkte-Noten-Schlüssel:

Note 1	=		100 – 92 Punkte	Note 2	=	unter	92 – 81 Punkte
Note 3	=	unter	81 – 67 Punkte	Note 4	=	unter	67 – 50 Punkte
Note 5	=	unter	50 – 30 Punkte	Note 6	=	unter	30 – 0 Punkte

Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2022 – Alle Rechte vorbehalten!

1. Aufgabe (25 Punkte)

a) 10 Punkte

```
void sort(raeume: Array von Raum)
      für i = 0, i < raeume.length-1; i++
            für j = 0; j < raeume.length – i -1; j++
                  wenn raeume[j].getBelegung() >
                              raeume[j+1].getBelegung()
                        Raum r = raeume[j]
                        raeume[j] = raeume[j+1]
                        raeume[j+1] = r
                  ende wenn
            ende für
      ende für
ende sort
```

b) 5 Punkte

```
vergleicheMitarbeiter(Mitarbeiter m1, Mitarbeiter m2): Integer
      diff: Double
      diff =m1.getGehalt() – m2.getGehalt()
      wenn diff == 0
            Rückgabe 0
      Ende wenn
      Wenn diff > 0
            Rückgabe 1
      Sonst
            Rückgabe -1
      ende wenn
ende vergleicheMitarbeiter
```

Hinweis: Eine Lösung mit Rückgabe m1.getGehalt() – m2.getGehalt() führt zu einem Typkonflikt zwischen Integer und Double. Eine Typkonvertierung von Gehalt zu Integer zu nicht richtigen Ergebnissen!

c) 3 Punkte

Eine generische Klasse ist eine typparametrisierte Klasse, das heißt Methoden und Variablen werden für einen vorgegebenen, aber beliebigen Typ definiert. Für den vorgegebenen Typ können Einschränkungen formuliert werden.

d) 7 Punkte

```
void sort(liste: List<T>, Function<T> f)
            für i = 0, i < liste.size()-1; i++
                  für j = 0; j < liste.size() – i -1; j++
                        wenn f(liste.get(j), liste.get(j+1)) > 0
                              T r = liste.get(j)
                              liste.set(j, liste.get(j+1))
                              liste.set(j+1, r)
                        ende wenn
                  ende für
            ende für
      ende sort
```

2. Aufgabe (25 Punkte)

aa) 4 Punkte

Testfall	Dezimalzahl	Erwartetes Ergebnis	Ergebnis
1	7	111	111
2	11	1011	1101

ab) 4 Punkte

Die Ausgabe „`Ausgabe(rest);`" erfolgt vor dem rekursiven Aufruf „`Umrechnung(dezimalzahl);`", beim Abstieg in die Rekursion. Daher ist das Ergebnis in der falschen Reihenfolge.
Korrektur: Ausgabe nach dem rekursiven Funktionsaufruf.

Korrekte Alternativen sind ebenfalls als richtig zu werten.

ac) 3 Punkte

Um Anweisungsüberdeckung zu erreichen, muss durch die Auswahl der Testfälle sichergestellt werden, dass jede Anweisung mindestens 1-mal ausgeführt wird.
Beim gegebenen Programm genügt ein Testfall.

b) 4 Punkte

z. B.:
Vorteil: Rekursive Lösungen sind zum passenden Problem einfacher oder eleganter zu schreiben.

Nachteil: Rekursive Lösungen benötigen mehr Speicher zur Laufzeit.

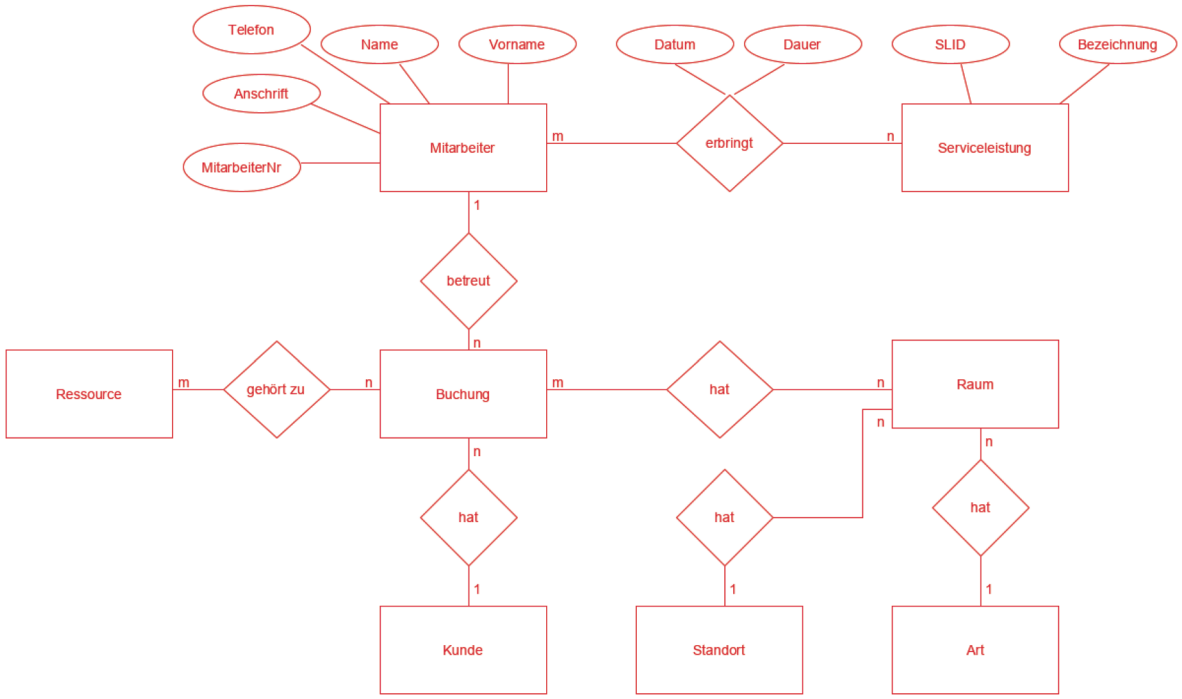
Korrekte Alternativen sind ebenfalls als richtig zu werten.

c) 10 Punkte

```
void Umrechnung(dezimalzahl: int)
BEGINN
      rest: int;
      i: int;
      List l();
      SOLANGE (dezimalzahl >0)
      BEGINN
            rest = dezimalzahl modulo 2;
            l.insert(rest);
            dezimalzahl = dezimalzahl div 2;
      ENDE
      ZÄHLE i VON l.size() BIS 0 SCHRITTWEITE -1
      BEGINN
            AUSGABE(l.element(i));
      ENDE
ENDE
```

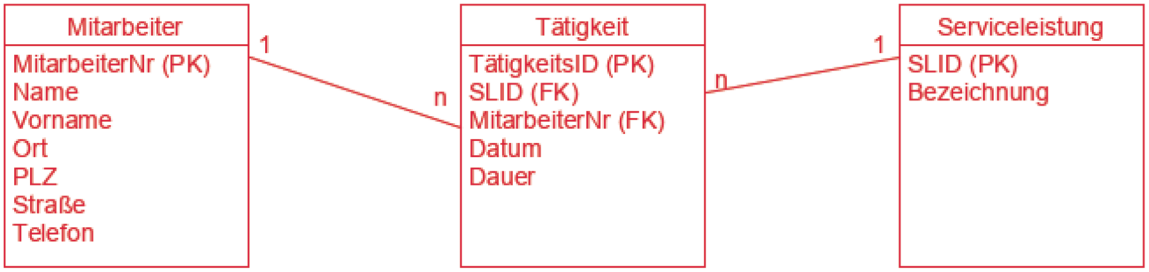
3. Aufgabe (25 Punkte)

a) 12 Punkte
1 Punkt für jeden richtigen Entitätstyp = 6 Punkte
1 Punkt für jede richtige Beziehung = 6 Punkte



Korrekte Alternativen sind ebenfalls als richtig zu werten.

b) 10 Punkte
Tabelle Mitarbeiter/Tätigkeit je 3 Punkte, Tabelle Serviceleistung 2 Punkte, je Beziehung 1 Punkt



Korrekte Alternativen sind ebenfalls als richtig zu werten.

c) 3 Punkte

Attribut	**Datentyp**
Dauer	Integer
PLZ	VarChar (String ,Text)
Bezeichnung	VarChar (String ,Text)

Korrekte Alternativen sind ebenfalls als richtig zu werten.

4. Aufgabe (25 Punkte)

a) 10 Punkte

```
SELECT Ma.Ma_ID AS ID
      ,Ma.Nachname
      ,Ma.Vorname
      ,Abt.Abteilung
      ,SUM(Fz.FehltageSum) AS Urlaubstage
      ,COUNT(Fz.Fz_ID) AS Urlaubseintraege
FROM Mitarbeiter AS Ma
LEFT JOIN Fehlzeit AS Fz ON Ma.Ma_ID = Fz.Ma_ID AND
                            Fz.FzA_ID = 2 AND
                            YEAR(Fz.VonDat) = 2022
LEFT JOIN Abteilung AS Abt ON Ma.Abt_ID = Abt.Abt_ID
GROUP BY Ma.Ma_ID, Ma.Nachname, Ma.Vorname, Abt.Abteilung;
```

b) 10 Punkte

```
SELECT Fz_ID
      ,Ma_ID
      ,FzA_ID
      ,VonDat
      ,BisDat
      ,FehltageSum
  INTO Fehlzeit_Archiv
  FROM Fehlzeit
  WHERE Ma_ID = 1;

DELETE
    FROM Fehlzeit
 WHERE Ma_ID = 1;
```

c) 5 Punkte

```
SELECT Ma.Ma_ID
      ,Ma.Nachname
      ,Ma.PLZ
      ,Ma.Ort
FROM Mitarbeiter AS Ma
WHERE Ma.PLZ LIKE '___3%'
```